SALMA20BCE7605

27/03/2021

DSA LAB Assignment

Firstly, the code:

Linked List Operations Implementation 〉 src 〉 com 〉 company 〉 Main.java 〉 Main 〉 main

Module | Main.java

```java
package com.company;
import java.util.*;
class Node {
    int data;
    Node next;
}
class Main{
    Node head;

    public void insertAtEnd(int data)
    {
        Node temp = new Node();
        temp.data = data;

        if(head == null)
        {
            head = temp;
        }
        else {
```

Linked List Operations Implementation 〉 src 〉 com 〉 company 〉 Main.java 〉 Main 〉 main

Main.java

```java
    public void insertAtStart(int data)
    {
        Node temp = new Node();
        temp.data = data;
        temp.next=head;
        head=temp;
    }

    boolean insertAfterNode(int data1, int data2)
    {
        Node d= findNode(data1);
        if(d==null) return false;
        Node temp = new Node();
        temp.data = data2;
        temp.next=d.next;
        d.next=temp;
        return true;

    }
```

Linked List Operations Implementation 〉 src 〉 com 〉 company 〉 Main.java 〉 Main 〉 main

Main.java

```java
    boolean deleteAtStart()
    {
        if(head==null) return false;
        head=head.next;
        return true;
    }

    boolean deleteAtEnd()
    {
        if(head==null) return false;
        if(head.next==null)
        {
            head=null;
            return true;
        }
        Node ptr=head;
        while(ptr.next.next!=null)
        {
            ptr=ptr.next;
```

Linked List Operations Implementation 〉 src 〉 com 〉 company 〉 Main.java 〉 Main 〉 main

Main.java

```java
boolean deleteAfterNode(int data)
{
    Node d=findNode(data);
    if(d==null||d.next==null) return false;
    d.next=d.next.next;
    return true;
}
Node findNode(int data)
{
    Node ptr=head;
    int c=0;
    while(ptr!=null)
    {
        if(ptr.data==data) return ptr;
        c++;
        ptr=ptr.next;
    }
    return null;
}
```

Linked List Operations Implementation 〉 src 〉 com 〉 company 〉 Main.java 〉 Main 〉 show

Main.java

```java
public void show()
{
    if(head == null)
    {
        System.out.println("Cannot print as list is empty! -.-");
    }else {
        Node p = head;
        while(p != null)
        {
            System.out.print(" "+p.data+"->");
            p = p.next;
        }
        System.out.println(" NULL.");
    }
}
public static void main(String[] args) {
    Main LL = new Main();
    int data1, data2;
    Scanner input = new Scanner(System.in);
```

Linked List Operations Implementation 〉 src 〉 com 〉 company 〉 Main.java 〉 Main 〉 show

Main.java

```java
    public void show()
    {
        if(head == null)
        {
            System.out.println("Cannot print as list is empty! -.-");
        }else {
            Node p = head;
            while(p != null)
            {
                System.out.print(" "+p.data+"->");
                p = p.next;
            }
            System.out.println(" NULL.");
        }
    }
    public static void main(String[] args) {
        Main LL = new Main();
        int data1, data2;
        Scanner input = new Scanner(System.in);
```

Linked List Operations Implementation 〉 src 〉 com 〉 company 〉 Main.java 〉 Main 〉 show

Main.java

```java
        Scanner input = new Scanner(System.in);
        System.out.println("\n----------------  SALMA 20BCE7605 wields her magi
        System.out.println("\nLet's start with the operations! Input your choic
        System.out.println("1. Go ahead and Traverse the current state of the s
                System.out.println("2. Insert at Start.");
        System.out.println("3. Insert at End.");
        System.out.println("4. Insert after Specified node.");

        System.out.println("5. Delete from the Start.");
        System.out.println("6. Delete at End.");
        System.out.println("7. Delete after specified node.");
        System.out.println("8. Exit");

        while(true)
        {
            System.out.println("Enter choice: ");
            int number = input.nextInt();

            switch(number) {
```

Linked List Operations Implementation 〉 src 〉 com 〉 company 〉 Main.java 〉 Main 〉 show

Main.java

```java
            switch(number) {
                case 1:
                    LL.show();
                    break;
                case 3:
                    System.out.println("Enter the data for the node that you war
                            data1 = input.nextInt();
                    LL.insertAtEnd(data1);
                    break;

                case 2:
                    System.out.println("Enter the data for the node that you war
                            data1 = input.nextInt();
                    LL.insertAtStart(data1);
                    break;
                case 4:
                    System.out.println("Enter the data for the node after which
                            data1 = input.nextInt();
                    data2 = input.nextInt();
```

Linked List Operations Implementation 〉 src 〉 com 〉 company 〉 Main.java 〉 Main 〉 show

Main.java

```java
                    break;
                case 7:
                    System.out.println("Enter the data for the node after which
                            data1 = input.nextInt();
                    if(LL.deleteAfterNode(data1))
                    {
                        System.out.println("Operation successful! *-*");
                    }
                    else
                    {
                        System.out.println("Operation failed. *.*");
                    }
                    break;
                case 8:
                    System.exit( status: 0);
                    break;
                default:
                    System.out.println("Beep. Beep. Invalid input. 0.0");
```

To be noted: The entire code isn't visible.

Now, The Fun part! ^.^

.

.

.

The Output:

```
---------------  SALMA 20BCE7605 wields her magic now with Singly Linked Lists! ^.^  --------------

Let's start with the operations! Input your choice, user.
1. Go ahead and Traverse the current state of the singly Linked List.
2. Insert at Start.
3. Insert at End.
4. Insert after Specified node.
5. Delete from the Start.
6. Delete at End.
7. Delete after specified node.
8. Exit
Enter choice:

2
Enter the data for the node that you want to insert at START.
1
Enter choice:
```

```
Enter choice:
3
Enter the data for the node that you want to insert at END.
3
Enter choice:
4
Enter the data for the node after which you want to insert and the data to insert.
1 2
Operation successful! *-*
Enter choice:
1
 1-> 2-> 3-> NULL.
Enter choice:
5
Operation successful! *-*
Enter choice:
6
Operation successful! *-*
Enter choice:
```

```
Enter choice:
2
Enter the data for the node that you want to insert at START.
33
Enter choice:
3
Enter the data for the node that you want to insert at END.
99
Enter choice:
1
 33-> 2-> 99-> NULL.
Enter choice:
7
Enter the data for the node after which you want delete
2
Operation successful! *-*
Enter choice:
1
 33-> 2-> NULL.
```

```
 33-> 2-> 99-> NULL.
Enter choice:
7
Enter the data for the node after which you want delete
2
Operation successful! *-*
Enter choice:
1
 33-> 2-> NULL.
Enter choice:
7
Enter the data for the node after which you want delete
2
Operation failed. *.*
Enter choice:
33
Beep. Beep. Invalid input. 0.0
```

P.S : This code was experimental.